USB ENGINEERING CHANGE NOTICE

Title: Interface Association Descriptors

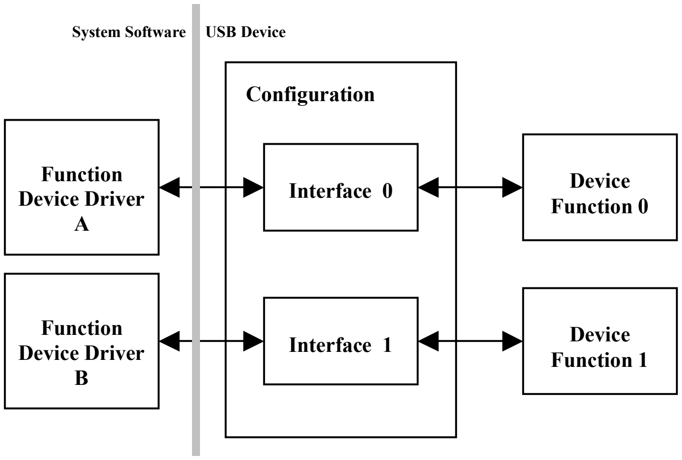
Applies to: Universal Serial Bus Specification, Revision 2.0

Summary of ECN

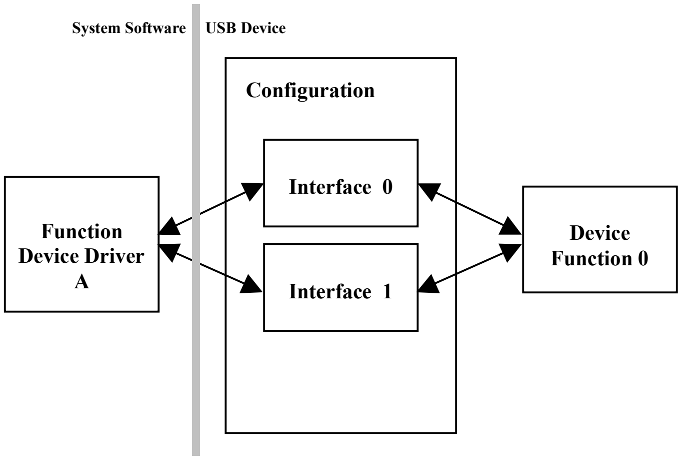
This ECN defines a new standard descriptor and interface numbering rules that allow a device to describe which interfaces are associated with the same device function. This allows the operating system to bind all of the appropriate interfaces to the same driver instance.

Reasons for ECN

The base configuration model assumed by the core USB framework was that there was always a 1:1 association between an interface and a function on a device. System software was designed to the intent of the core specification and assumes one driver per function (and one interface) (see figure below).



Several device class specifications have exceeded the core USB specification framework and defined device functions that utilize multiple interfaces (i.e. multiple interface descriptors). The model to support this still requires only one function driver per function, but also requires multiple interfaces getting bound to the same driver instance (see figure below). Unfortunately, there is no standard method to allow a device, via the device framework, to describe which interfaces in a configuration should be associated with the same function.



This change notice defines the necessary extensions to the device framework that allow the device to annotate which interfaces are associated with the same function. This device framework extension will be eventually required for all devices that utilize multiple interfaces per device function.

Impact on Existing Peripherals and Systems:

No Impact.

Hardware Implications:

None.

Software Implications:

No impact to existing operating system versions and existing device classes. The new descriptors are ignored by system software. The interfaces they describe don't change.

This feature must be supported by future implementations of devices that use multiple interfaces to manage a single device function.

There exists an impact to **future/new** device implementations for device classes that are not currently supported by the OS. Specifically, if a device implementation includes multiple functional units (with multiple interfaces per unit) then the device will only be correctly enumerated on an OS implementation that supports this new descriptor.

In order to more easily enhance existing OS implementations with the capability to handle devices that use this descriptor, a device class code will be allocated with the intent that all devices that use the interface association descriptor will use this class code in their device descriptor. This will allow easy installation of a new driver that knows how to parse and enumerate configurations that include the interface association descriptor. The class code for the IAD will be documented on the usb.org website.

It is under the responsibility of the existing Device Class working groups to determine whether their individual specifications need to be modified to work with or take advantage of this new framework extension.

Compliance Testing Implications:

The standard compliance toolset (USBCV) must be eventually updated to check the format (and use) of these new descriptors. In addition, some rules must be established for the compliance tools to determine which device should be using these descriptors and fail them for not using them.

Specification Changes

The following proposal is backward compatible with previous operating systems. It leaves current interface definitions alone and adds a new descriptor type. For older operating systems versions, the new descriptors will be ignored and the old mechanisms will prevail. For new operating system versions, the new descriptors will be in effect.

Add the following to table 9-6 (note, the core specification currently defines values 1-8. Since publication, values 9 & 10 have been allocated as noted below).

Descriptor Types	Value
DEVICE	1
CONFIGURATION	2
STRING	3
INTERFACE	4
ENDPOINT	5
DEVICE_QUALIFIER	6
OTHER_SPEED_CONFIGURATION	7
INTERFACE_POWER	8
OTG	9
DEBUG	10
INTERFACE_ASSOCIATION	11

The following is the definition of an Interface Association Descriptor. At the next spec revision update, it should be included as a new section 9.6.6 (moving the current section 9.6.6 to 9.6.7, and so-on).

9.X.Y Interface Association

The Interface Association Descriptor is used to describe that two or more interfaces are associated to the same function. An 'association' includes two or more interfaces and all of their alternate setting interfaces. A device must use an Interface Association descriptor for each device function that requires more than one interface. An Interface Association descriptor is always returned as part of the configuration information returned by a GetDescriptor(Configuration) request. An interface association descriptor cannot be directly accessed with a GetDescriptor() or SetDescriptor() request. An interface association descriptor must be located before the set of interface descriptors (including all alternate settings) for the interfaces it associates. All of the interface numbers in the set of associated interfaces must be contiguous. Table 9-Z shows the standard interface association descriptor. The interface association descriptor includes function class, subclass and protocol fields. The values in these fields can be the same as the interface class, subclass and protocol values from any one of the associated interfaces. The preferred implementation, for existing device classes, is to use the interface class, subclass and protocol field values from the first interface in the list of associated interfaces.

Table 9–Z. Standard Interface Association Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes.
1	*bDescriptorType*	1	Constant	INTERFACE ASSOCIATION Descriptor.
2	*bFirstInterface*	1	Number	Interface number of the first interface that is associated with this function.
3	*bInterfaceCount*	1	Number	Number of contiguous interfaces that are associated with this function.
4	*bFunctionClass*	1	Class	Class code (assigned by USB-IF). A value of zero is not allowed in this descriptor. If this field is FFH, the function class is vendor-specific. All other values are reserved for assignment by the USB-IF.
5	*bFunctionSubClass*	1	SubClass	Subclass code (assigned by USB-IF). If the *bFunctionClass* field is not set to FFH all values are reserved for assignment by the USB-IF.
6	*bFunctionProtocol*	1	Protocol	Protocol code (assigned by USB-IF). These codes are qualified by the values of the *bFunctionClass* and *bFunctionSubClass* fields.
7	*iFunction*	1	Index	Index of string descriptor describing this function.

Note: Since this particular feature was not included in earlier versions of the USB specification, there is issue with how existing USB OS implementations will support devices that use this descriptor. It is strongly recommended that device implementations utilizing the *interface association descriptor* use the Multi-interface

Function Class codes in the device descriptor. This allows simple and easy identification of these devices and allows on some operating systems, installation of an upgrade driver that can parse and enumerate configurations that include the Interface Association Descriptor. The Multi-interface Function Class code is documented on the http://www.usb.org/developers/docs website.